What is claimed is:

1. A software program for providing instructions to one or more processors to execute processes on an embedded computing device configured for establishing a network connection with at least one other computing device, comprising:

(a) an operating system layer;

(b) an application framework; and

(c) a programming environment including a contention locking scheme for setting light object locks, which are handled in user space, and heavy object locks, which are handled at the system level, and wherein the contention locking scheme is configured to set a light object lock on an initially unlocked object when a first thread attempts to lock the object, and to maintain a light lock on the object when a nested intra-thread lock is attempted by the first thread.

2. The software program of Claim 1, wherein the contention locking scheme is further configured to set a heavy object lock on the object when a second thread attempts to lock the object while the object is lightly locked by the lock attempt by the first thread.

3. The software program of Claim 2, wherein the contention locking scheme includes a stack-based local lock slot structure for addressing stack variables to identify threads.

4. The software program of Claim 3, wherein a first stack corresponds to a data area of the first thread and a second stack corresponds to a data area of the second thread, and wherein the first and second stacks are separated by at least a reserved area which is set at the end of each stack.

5. The software program of Claim 4, wherein the contention locking scheme is configured to maintain the light lock when an address difference between a current lock slot of the first thread for the lightly locked object and that of the nested intra-thread locking attempt is determined to be less than the reserved area.

6. The software program of Claim 4, wherein the contention locking scheme is configured to set the heavy lock when an address difference between a current lock slot of the first thread for the lightly locked object and that of the locking attempt by the second thread is determined to be greater than the reserved area.

7. The software program of Claim 3, wherein the contention locking scheme further includes a lock structure and a lock structure reference in an object header of the object, the lock structure including a lock holder and wait queues.

8. The software program of Claim 1, wherein the contention locking scheme includes a stack-based local lock slot structure for addressing stack variables to identify threads.

9. The software program of Claim 8, wherein the contention locking scheme is configured to maintain the light lock when an address difference between a current lock slot of the first thread for the lightly locked object and that of the nested intra-

thread locking attempt is determined to be less than a reserved area which is set at the end of each stack.

10. A software program for providing instructions to one or more processors to execute processes on an embedded computing device configured for establishing a network connection with at least one other computing device, comprising:

(a) an operating system layer;

(b) an application framework; and

(c) a programming environment including a contention locking scheme for setting light object locks, which are handled in user space, and heavy object locks, which are handled at the system level, and wherein the contention locking scheme is configured to set a light object lock on an initially unlocked object when a first thread attempts to lock the object, and to maintain a light lock on the object when a nested intra-thread lock is attempted by the first thread, and

(d) wherein the contention locking scheme includes a stack-based local lock slot structure for addressing stack variables to identify threads, and

(e) wherein the contention locking scheme further includes a lock structure and a lock structure reference in an object header of the object, the lock structure including a lock holder and wait queues.

11. A method for executing processes with contention-locking which uses light locks, which are handled at the user level, and heavy locks, which are handled at the system level, on an embedded computing device configured for establishing a network connection with at least one other computing device, comprising the steps of:

(a) setting a light object lock, for handling at the user level, on an initially unlocked object when a first thread attempts to lock the object; and

(b) determining to maintain the light object lock when a nested intra-thread lock is attempted by the first thread.

12. The method of Claim 11, further comprising the step of setting a heavy object lock on the object when a second thread attempts to lock the object while the object is lightly locked by the lock attempt by the first thread.

13. The method of Claim 12, further comprising the step of addressing stack variables to identify threads using a stack-based local lock slot structure.

14. The method of Claim 13, wherein the addressing step includes addressing the first thread at a first stack and addressing the second thread at a second stack, and wherein the first and second stacks are separated by at least a reserved area which is set at the end of each stack.

15. The method of Claim 14, wherein the step of determining to maintain the light object lock includes determining that an address difference between a current lock slot

of the first thread for the lightly locked object and that of the nested intra-thread locking attempt is less than the reserved area.

16. The method of Claim 13, further comprising the step of forming a lock structure and a lock structure reference in an object header of the object, the lock structure including a lock holder and wait queues.

17. The method of Claim 12, wherein the step of determining to set the heavy lock includes determining that an address difference between a current lock slot of the first thread for the lightly locked object and that of the locking attempt by the second thread is greater than the reserved area.

18. The method of Claim 11, further comprising the step of addressing stack variables to identify threads using a stack-based local lock slot structure.

19. The method of Claim 18, wherein the step of determining to maintain the light object lock includes determining that an address difference between a current lock slot of the first thread for the lightly locked object and that of the nested intra-thread locking attempt is less than the reserved area.

20. The method of Claim 18, further comprising the step of forming a lock structure and a lock structure reference in an object header of the object, the lock structure including a lock holder and wait queues.